AC/2178

TRANSMITTAL OF APPEAL BRIEF (Large Entity)

Docket No.
END920000080US1

Re Application Of: **James R. Wason**

Application No.	Filing Date	Examiner	Customer No.	Group Art Unit	Confirmation No.
09/616,809	**July 14, 2000**	**Joshua d. Campbell**	**23389**	**2178**	**6597**

Invention: **TEXT FILE INTERFACE SUPPORT IN AN OBJECT ORIENTED APPLICATION**

COMMISSIONER FOR PATENTS:

Transmitted herewith is the Appeal Brief in this application, with respect to the Notice of Appeal filed on:
March 23, 2006

The fee for filing this Appeal Brief is: **$500.00**

☐ A check in the amount of the fee is enclosed.

☒ The Director has already been authorized to charge fees in this application to a Deposit Account.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. **09-0457/IBM**. I have enclosed a duplicate copy of this sheet.

☐ Payment by credit card. Form PTO-2038 is attached.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

John S Sensny
Signature

Dated: **June 29, 2006**

John S. Sensny
Registration No. 28,757

Scully, Scott, Murphy & Presser, P.C.
400 Garden City Plaza - Suite 300
Garden City, New York 11530
(516) 742-4343

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" [37 CFR 1.8(a)] on
June 29, 2006.
(Date)

John S Sensny
Signature of Person Mailing Correspondence

John S. Sensny
Typed or Printed Name of Person Mailing Correspondence

cc: **JSS:jy**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: James R. Wason

Examiner: Joshua D. Campbell

Serial No: 09/616,809

Art Unit: 2178

Filed: July 14, 2000

Docket: END920000080US1 (13679)

For: TEXT FILE INTERFACE SUPPORT IN AN OBJECT ORIENTED APPLICATION

Dated: June 29, 2006

Confirmation No.: 6597

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

Pursuant to 35 U.S.C. 134 and 37 C.F.R. 41.37, entry of this Appeal Brief in support of the Notice of Appeal filed April 3, 2006, in the above-identified matter is respectfully requested.

CERTIFICATE OF MAILING UNDER 37 C.F.R. §1.8(a)

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on date shown below.

Dated: June 29, 2006

John S. Sensny

I. Statement of Real Party in Interest

The real party in interest in the above-identified patent application is the International Business Machines Corporation.

II. Statement of Related Appeals and Interferences

There are no other prior or pending appeals, interferences or judicial proceedings known to appellants, the appellants' legal representative, or assignee which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. Status of Claims

A. Claim Status

Claim 1 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 2 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 3 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 5 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 6 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 7 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 8 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 9 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 10 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 11 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 12 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 13 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 14 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 15 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 16 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

Claim 17 stands rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent 6,317,871 (Andrews, et al.).

B. Appealed Claims

Claims 1-3 and 5-17 are appealed. A clean copy of these claims is contained in Appendix A to this Appeal Brief.

IV. Status of Amendments

No Amendments have been filed subsequent to the final rejection. A response, dated February 27, 2006, was filed. That response has been considered by the Examiner and the Examiner has held that this response does not place the application in condition for allowance.

V. Summary of Claimed Subject Matter

Independent Claims 1, 6 and 10 are involved in this appeal.

Concise explanation of the subject matter of claim 1

Claim 1 is directed to a method of processing a text file in a computer application. This method comprises the steps of forming (page 3, line 14; page 5, lines 26 and 27; page 9, lines 8-10; page 11, lines 9-16; page 15, lines 21-32; page 20, line 31; page 23, line 18; Figure 1) a plurality of templates having literal fragments of the text file, and providing (page 3, lines 17 and 18; page 6, lines 9-19; page 33, line 6; page 35, line 15; Figure 1) a macro class to map data from the text file to the computer application. The method comprises the further steps of embedding (page 3, lines 18 and 19; page 6, lines 21-23; page 9, lines 28-32; page 12, lines 1 and 2; page 32 line 26; page 33, line 5; Figure 1) in one of the templates a pointer to the macro class, and using (page 3, line 15; page 5, lines 26 and 27; page 20, line 16; page 21, line 24; page 27, line 15; page 29, line 12; Figure 1) said one of the templates as an overlay to parse the text file into segments having data, or as a prototype to generate a segment of an output file. This using step includes the steps of i) reaching (page 3, lines 19 and 20; page 6, line 24; page 16, lines 7 and 8; Figure 1) said pointer in said one of the templates, ii) when said pointer is reached, using said pointer to invoke said macro class and using said macro class to map data from one of the segments of the text file

to the computer application, and iii) said macro class then invoking (page 6, lines 25-27; page 16, lines 14 and 15; page 17, lines 23 and 24) another one of the templates to further process the text file.

Concise explanation of the subject matter of Claim 6

Claim 6 defines a system for processing a text file in a computer application. This system comprises means forming (page 3, line 14; page 5, lines 26 and 27; page 9, lines 8-10; page 11, lines 9-16; page 15, lines 21-32; page 20, line 31; page 23, line 18; Figure 1) a plurality of templates having literal fragments of the text file; means forming (page 3, lines 18 and 19; page 6, lines 21-23; page 9, lines 28-32; page 12, lines 1 and 2; page 32, line 26; page 33, line 5; Figure 1) a macro class to map data from the text file to the computer application, wherein a pointer to the macro class is embedded in one of the templates; and means for using (page 3, line 15, page 5, lines 26 and 27; page 20, line 16; page 21, line 24; page 27, line 15; page 29, line 12; Figure 1) said one of the templates as an overlay to parse the text file into segments having data, or as a prototype to generate a segment of an output file. As described in Claim 6, the means for using includes i) means for using (page 3, line 20; page 6, line 21; page 16, lines 7 and 8; page 17, line 23; page 3, lines 22-25; page 6, lines 9-19; page 12, lines 1-6, Figure 1) said pointer, when said pointer is reached in said one of the templates, to invoke said macros class, ii) means to use (page 3, line 20; page 6, line 24; page 16, lines 7 and 8; page 17, line 33; page 3, lines 22-25; page 6, lines 9-19; page 12, lines 1-6) said macro class to map data from one of the segments of the text file to the computer application; and iii) means to use (page 6, lines 25-27; page 16, lines 14 and 15; page 17, lines 23 and 24) the macro class to invoke another one of the templates to further process the text file.

Concise explanation of the subject matter of Claim 10

Independent Claim 10 is directed to a program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for processing a text file in a computer application. These method steps comprise forming (page 3, line 14; page 4, lines 26 and 27; page 9, lines 8-10, page 11, lines 9-16; page 15, lines 21-30; page 20, line 31; page 23, line 18; Figure 1) a plurality of templates having literal fragments of the text file, providing (page 3, lines 17 and 18; page 6, lines 9-19; page 33, line 6; page 35, line 15, Figure 1) a macros class to map data from the text file to

the computer application, embedding (page 3, lines 18 and 19; page 6, lines 21-23; page 9, lines 28-32; page 12, lines 1 and 2; page 32, lines 26; page 33, line 5; Figure 1) in one of the templates a pointer to the macro class; and using (page 3, line 15; page 5, lines 26 and 27; page 20, line 16; page 21, line 24; page 27, line 15; page 29, line 12; Figure 1) said one of the templates as an overlay to parse the text file into segments having data, or as a prototype to generate a segment of an output file. This using step, as described in Claim 10 includes the steps of i) reaching (page 3, lines 19 and 20; page 6, line 24; page 16, lines 7 and 8; Figure 1) said pointer in said one of the templates, ii) when said pointer is reached, using (page 3, line 20; page 6, line 21; page 16, lines 7 and 8; page 17, line 23; page 3, lines 22-25; page 6, lines 9-19; page 12, lines 1-6; Figure 1) said pointer to invoke said macro class and using said macro class to map data from one of the segments of the text file to the computer application, and iii) said macro class then invoking (page 6, lines 25-27; page 16, lines 14 and 15; page 17, lines 23 and 24) another one of the templates to further process the text file.

Identification of means plus function and step plus function under 35 U.S.C. 112, and identification of the structure, materials or acts described in the specification as corresponding to each claimed function.

Claim 1

Claim 1 includes the following step plus function clauses:

1. forming a plurality of templates having literal fragments of the text file
corresponding structure, material or acts

This is done on the Enterprise Application Development Platform (EADP) (Page 4, lines 11-13) by taking fragments of text that include the literals for a text str5eam (page 9, lines 8 and 9). The template is a piece of text with embedded keywords (page 14, lines 21 and 22), (page 5, lines 26 and 27), (page 3, line 14), (page 11, lines 9-16), (page 15, lines 21-32), (page 22, line 31 to page 23, line 18).

2. providing a macro class to map data from the text file to the computer application
corresponding structure, material or acts

This is done on the EADP (page 4, lines 11-13). The invention provides a basic set of macros and the facilities to add more (page 6, lines 17-19). A macro is provided that supports the navigation of a complex object structure (page 11, lines 20 and 21). Macros needed for a particular file structure can be added as needed (page 42, lines 30-32).

3. embedding in one of the templates a pointer to the macro class

corresponding structure, material or acts

This is done by the EADP (page 4, lines 11-13) by embedding a keyword into the template that points to the macro class (page 3, lines 18 and 19), (page 6, lines 21-23), (page 9, lines 28-32), (page 12, lines 1 and 2), (page 32, lines 26 to page 33, line 5).

4. using said one of the templates as an overlay to parse the text file into segments having data, or as a prototype to generate a segment of an output file

corresponding structure, material or acts

This is done by the EADP (page 4, lines 11-13) by using the template as a mask to parse the text file. The text is broken up into lines (page 13, line 15; page 5, lines 26 and 27; page 20, line 16; page 21, line 24; page 27, line 15; page 29, line 12).

5. reaching said pointer in said one of the templates

corresponding structure, material or acts

This is done by the EADP (page 4, lines 11-13). As the template is used, a keyword in the template is reached, and this keyword calls the macro class (page 3, lines 19 and 20; page 6, line 24; page 16, lines 7 and 8).

6. when said pointer is reached, using said pointer to invoke said macro class and using said macro class to map data from one of the segments of the text file to the computer application

corresponding structure, material or acts

This is done by the EADP (page 4, lines 11-13). The keyword in the template points to the macro, and the macro is called to further process the text file. Them across specify points in the input stream from which data are to be taken and what to do with that data. The macros can also specify that a field is of a fixed length, so that the number of bytes is automatically advanced in the stream (page 3, line 20; page 6, line 24; page 16, lines 7 and 9; page 17, line 23; page 3, lines 22-25; page 6, lines 9-19; page 12, lines 1-6).

7. said macro class then invoking another one of the templates to further process the text file
corresponding structure, material or acts

This is done by the EADP (page 4, lines 11-13). The macros is given the name of another template, and as part of the processing of the macro, the macro can invoke that other template (page 6, lines 25-27, page 16, lines 14 and 15; page 17, lines 23 and 24).

Claim 6

Claim 6 includes the following means plus function clauses

1. means forming a plurality of templates having literal fragments of the text file
corresponding structure, material or acts

This is done on the Enterprise Application Development Platform (EADP) (page 4, lines 11-13) by taking fragments of text that include the literals for a text stream (page 9, lines 8 and 9). The template is a piece of text with embedded keywords (page 15, lines 21 and 22), (page 5, lines 26 and 27); (page 3, line 14); (page 11, lines 9-16); (page 15, lines 21-30); (page 22, line 31 to page 23, line 18).

2. means forming a macro class to map data from the text file to the computer application, wherein a pointer to the macro class is embedded in one of the templates
corresponding structure, material or acts

This is done on the EADP (page 4, lines 11-13). The invention provides a basic set of macros and the facilities to add more (page 6, lines 17-19). A macro is provided that supports the navigation of a complex object structure (page 11, lines 20 and 21). Macros needed for a particular file structure can be added as needed (page 42, lines 30-32). A pointer is embedded in the template by embedding a keyword into the template that points to the macro class (page 3, lines 18 and 19), (page 6, lines 21-23), (page 9, lines 28-32), (page 12, lines 1 and 2), (page 32, line 26 to page 33, line 5).

3. means for using said one of the templates as an overlay to parse the text file into segments having data, or as a prototype to generate a segment of an output file
corresponding structure, material or acts

This is done on the EADP (page 4, lines 11-13) by embedding a keyword into the template that points to the macros class (page 3, lines 18 and 19), (page 6, lines 21-23), (page 9, lines 28-32), (page 12, lines 1 and 2), (page 32, line 26 to page 33, line 5).

4. means for using said pointer, when said pointer is reached in said one of the templates, to invoke said macros class
corresponding structure, material or acts

This is done on the EADP (page 4, lines 11-13) by using the template as a mask to parse the text file. The text is broken up into lines (page 3, line 15, page 5, lines 26 and 27; page 20, line 16 to page 21, line 24; page 27, line 15 to page 29, line 12).

5. means to use said macro class to map data from one of the segments of the text file to the computer application
corresponding structure, material or acts

This is done by the EADP (page 4, lines 11-13). As the template is used, a keyword in the template is reached, and this keyword calls the macro class (page 3, lines 19 and 20; page 6, line 24; page 16, lines 7 and 8).

6. means to use the macro class to invoke another one of the templates to further process the text file
corresponding structure, material or acts

This is done by the EADP (page 4, lines 11-13). The keyword in the template points to the macro, and the macro is called to further process the text file. Them across specify points in the input stream from which data are to be taken and what to do with that data. The macros can also specify that a field is of a fixed length, so that the number of bytes is automatically advanced in the stream (page 3, line 20; page 6, line 24; page 16, lines 7 and 9; page 17, line 23; page 3, lines 22-25; page 6, lines 9-19; page 12, lines 1-6).

7. said macro class then invoking another one of the templates to further process the text file
corresponding structure, material or act

This is done by the EADP (page 4, lines 11-13). The macros is given the name of another template, and as part of the processing of the macro, the macro can invoke that other template (page 6, lines 25-27, page 16, lines 14 and 15; page 17, lines 23 and 24).

In addition, Claims 14-16 are argued separately.

Claim 14 includes the following step plus function clauses:

1. passing to said macro class a name for said another template when said macro class is invoked
corresponding structure, material or acts

This is done on the EADS by passing to the template the name of the next template to process (page 16, lines 14-19). This may be done as part of the invocation of the macro (page 6, line 25-26).

2. said macro class using said name to invoke said another template to further process the text file
corresponding structure, material or acts

This is done on the EADS. As part of the macro processing, the macro invokes the template whose name it has been given (page 6, lines 25-27).

Claim 16 includes the following step plus function clause:

said macro class using said name to invoke said another template to further process the text file

corresponding structure, material or acts

This is done on the EADS. As part of the macro processing, the macro invokes the template whose name it has been given (page 6, lines 25-27).

VI. Grounds of Rejection to be Reviewed On Appeal

Appellants ask that the following grounds be reviewed:

1. Whether Claims 1-3 and 5-17 are fully anticipated, under 35 USC §102, by Andrews, et al.

2. Whether Claims 14-16 are fully anticipated, under 35 U.S.C. 102, by Andrews, et al.

VII. Argument

The rejection of Claims 1-3, 5-17 is respectfully traversed because Andrews clearly neither discloses nor suggests using a macro, which was invoked by one template, both (i) to map data from a text file to a computer application, and (ii) itself to invoke another template to further process the text file, as described in independent Claims 1, 6 and 11. Claim 1 is representative of Claims 6 and 11.

Andrews, et al.

Andrews, et al. discloses a procedure for translating source code from one high-level computer language to another. In the disclosed procedure, fragment templates and partition templates are extracted from a source language text file, and a check is made for textual consistency of the target language code

generated for each partition template. The described process then pieces together a target language code file from the partition templates, and combines pieces of the target language code file that were generated in different translation sessions.

One specifically disclosed translation process is referred to as the Rosetta Translator. This Translator uses a syntax tree representation and a token mechanism. A source language syntax tree is used to represent the syntactic structure of a virtual source, and a source language fragment tree is employed to represent the virtual source production mechanisms that were used to create the virtual source.

Differences Between the Claims and Andrews, et al.

Andrews, et al. does not disclose using a macro, which was invoked by one template, both (i) to map data from a text file to a computer application, and (ii) itself to invoke another template to further process the text file.

Discussion

In rejecting the claims, the Examiner (see Office Action of December 23, 2005, pages 2 and 3, paragraph 4) specifically cited the portion of Andrews, et al. from Column 7, Line 65 to Column 9, Line 50. This portion of Andrews, et al. discusses how the partition templates are used and how the Rosetta Translator builds a pTAL fragment tree, translates the contents of a macro body, and pieces together instances of target language partition templates to form target language output files.

Applicant's Attorney has carefully reviewed Andrews, et al, particularly columns 7-9. In the process described in Andrews, et al, macros are translated, but these macros are not used to map the translation. Andrews, et al. includes several references to translating macros. For example, in Column 8, Lines 41-43, Andrews, et al. notes that "the text of macro actual parameter fragment templates is collapsed into the invoking partition just before the file is pieced together.

In Column 8, Lines 47-50 of Andrews, et al., it is explained that the source generator "fits the text representing the body of the macro stuff into the macro definition." Further, in Column 8, Lines 57-59 of

Andrews, et al., it is explained that "Inconsistently translated code can appear in any partition, not just in macro bodies." Also, in Column 9, Lines 17-19, Andrews, et al. refer to "exposing different code (which might contain macro definitions) and directives to the translator." Nowhere in Andrews, et al, though, is there any reference to using the macro to map the translation.

Significantly, the Court of Appeals for the Federal Circuit emphasizes that a strict identity test must be met in order for a reference to anticipate a claim under 35 U.S.C. 102. For instance, in Apple Computer, Inc. v. Articulate Systems, Inc., 57 USPQ2d 1057, 1061 (Fed. Cir 2000), the Court explained that: "Anticipation under 35 U.S.C. 102 requires the disclosure in a single piece of prior art of each and every limitation of a claimed invention." "Substantial identity" or "equivalency" is not sufficient. RCA Corp. V. Applied Digital Data Sys., Inc., 221 USPQ 385 (Fed. Cir. 1984).

Independent Claims 1, 6 and 10 of the present application describe how the macros are used in the instant invention. Each of these claims describes the features that a plurality of templates are formed, and that during processing of the text file by one of the templates, a pointer in that template is used to invoke the macro class. These claims also describe the features that this macro class maps data from the text file to the computer application, and then itself is used to invoke another one of the templates to further process the text file. The way in which the first template invokes a macro, which then invokes a second template, is not shown in Andrews, et al.

This feature of the invention is of considerable utility. This nested use of templates and macros allows a processing structure to be built up that mirrors the inherent structure of the text file. Since the behavior of the macro depends both on its internal logic and the template it is passed to invoke, it is possible to reuse the same macro to do different things by passing it a different template. The net effect is that the bulk of the logic needed to describe flow of control may be included in the template structure.

Claims 14-16

Claims 14, 15 and 16 are dependent from Claims 1, 6 and 10, respectively, and distinguish therewith over Andrews, et al. Further, Claims 14-16 distinguish over Andrews, et al. because of the additional features described in Claims 14-16. Claim 14 is representative of Claims 15 and 16.

Claim 14 elaborates on the procedure used for a macro to invoke another template. In particular, as described in Claim 14, the name of another template is passed to the macro, and the macro then uses that name to invoke that other template. This results in a nested aggregation of templates and macros.

Andrews, et al, since it does not show using the macro to invoke another template, clearly does not show or suggest any specific procedure, such as the one set forth in Claim 14, for actually invoking that other template. Accordingly, Claim 14 and Claims 15 and 16 independently distinguish over Andrews, et al. because of the features expressly set forth in these claims.

Conclusion

Because of the above-discussed differences between Claims 1, 6 and 10 and Andrews, et al, it cannot be said that this reference anticipates any of these claims. Claims 2, 3, 5, 14 and 17 are dependent from, and distinguish over Andrews, et al, with, Claim 1. Claims 7-9 and 15 are dependent from Claim 6 and distinguish therewith over Andrews, et al, and Claims 11-13 and 16 are dependent from Claim 10 and distinguish therewith over Andrews, et al. In addition, Claims 14-16 separately distinguish over Andrews, et al. because of the additional features expressly described in these claims. Thus, the rejection of the claims over Andrews, et al, under 35 U.S.C. 102, is not proper, and the Board is respectfully requested to reverse this rejection.

VII. Claims Appendix

A clean copy of Claims 1-3 and 5-17 is contained in Appendix A to this Appeal Brief.

IX. Evidence Appendix

Appellants are not relying on any affidavits, extrinsic documents or extrinsic evidence.

X. Related Proceedings Appendix

As indicated above, there are no other prior or pending appeals, interferences or judicial proceedings known to appellants, the appellants' legal representative, or assignee which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

Respectfully submitted,

Dated: June 29, 2006

John S. Sensny
Registration No. 28,757
Attorney for Applicants

Scully, Scott, Murphy & Presser, P.C.
400 Garden City Plaza – Suite 300
Garden City, New York 11530
(516) 742-4343

JSS:jy
Enclosure: Appendix A

APPENDIX A

1. A method of processing a text file in a computer application, comprising the steps:

forming a plurality of templates having literal fragments of the text file;

providing a macro class to map data from the text file to the computer application;

embedding in one of the templates a pointer to the macro class; and

using said one of the templates as an overlay to parse the text file into segments having data, or as a prototype to generate a segment of an output file;

said using step including the steps of:

i) reaching said pointer in said one of the templates,

ii) when said pointer is reached, using said pointer to invoke said macro class and using said macro class to map data from one of the segments of the text file to the computer application; and

iii) said macro class then invoking another one of the templates to further process the text file.

2. A method according to Claim 1, wherein the macro class reads in a segment of the text file and uses the segment to initiate application update processing.

3. A method according to Claim 1, wherein the macro class derives data from the application and formats it into the text file.

5. A method according to Claim 1, further comprising the step of providing an interface controller to prevent structure clashes by placing text data into appropriate places in a complex object structure as the text file is processed.

6. A system for processing a text file in a computer application, comprising:

means forming a plurality of templates having literal fragments of the text file;

means forming a macro class to map data from the text file to the computer application, wherein a pointer to the macro class is embedded in one of the templates;

means for using said one of the templates as an overlay to parse the text file into segments having data, or as a prototype to generate a segment of an output file;

said means for using including

i) means for using said pointer, when said pointer is reached in said one of the templates, to invoke said macros class,

ii) means to use said macro class to map data from one of the segments of the text file to the computer application;

iii) means to use the macro class to invoke another one of the templates to further process the text file.

7. A system according to Claim 6, wherein the macro class reads in a segment of the text file and uses the segment to initiate application update processing.

8. A system according to Claim 6, wherein the macro class derives data from the application and formats it into the text file.

9. A system according to Claim 6, further comprising an interface controller to prevent structure clashes by placing text data into appropriate places in a complex object structure as the text file is processed.

10. A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for processing a text file in a computer application, said method steps comprising:

forming a plurality of templates having literal fragments of the text file;

providing a macros class to map data from the text file to the computer application;

embedding in one of the templates a pointer to the macro class; and

using said one of the templates as an overlay to parse the text file into segments having data, or as a prototype to generate a segment of an output file;

said using step including the steps of:

i) reaching said pointer in said one of the templates,

ii) when said pointer is reached, using said pointer to invoke said macro class and using said macro class to map data from one of the segments of the text file to the computer application;

iii) said macro class then invoking another one of the templates to further process the text file.

11. A program storage device according to Claim 10, wherein the macro class reads in a segment of the text file and uses the segment to initiate application update processing.

12. A program storage device according to Claim 10, wherein the macro class derives data from the application and formats it into the text file.

13. A program storage device according to Claim 10, wherein said method steps further comprise the step of providing an interface controller to prevent structure clashes by placing text data into appropriate places in a complex object structure as the text file is processed.

14. A method according to Claim 1, wherein:

the step of using said pointer to invoke said macro class includes the step of passing to said macro class a name for said another template when said macro class is invoked; and

the step of using said macro class includes the step of, said macro class using said name to invoke said another template to further process the text file.

15. A system according to Claim 6, wherein:

when the macro class is invoked, a name for said another template is passed to the macro class from said template; and

said macro class uses said name to invoke said another template to further process the text file.

16. A program storage device according to Claim 10, wherein

when the macro class is invoked, a name for said another template is passed to the macro class from said template; and

the step of using said macro class includes the steps of, said macro class using said name to invoke said another template to further process the text file.

17. A method according to Claim 5, wherein the step of providing the interface controller includes the steps of:

using the interface controller to set up said complex object structure and to place said text data into said object structure as the text file is processed; and

after the entire text file is processed, using said structure to process updating data into said application.